

On Solving Convex Optimization Problems with Linear Ascending Constraints

Zizhuo Wang*

January 1, 2013

Abstract

We propose two algorithms for solving convex optimization problems with linear ascending constraints. When the objective function is separable, we propose a dual method which terminates in a finite number of iterations. In particular, the worst case complexity of our dual method improves over the best-known result for this problem in [2]. We then propose a gradient projection method to solve a more general class of problems. The gradient projection method uses the dual method as a subroutine in each projection step and does not need to evaluate the inverse gradient functions as most dual method do. Numerical experiments show that both our algorithms work very well in test problems.

1 Introduction

In this paper, we consider the following optimization problem:

$$(\mathbf{P1}) \quad \text{minimize}_{\vec{y}} \quad F(\vec{y}) = f(y_1, \dots, y_n) \quad (1)$$

$$\text{subject to} \quad \sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n-1 \quad (2)$$

$$\sum_{i=1}^n y_i = (\leq) \sum_{i=1}^n \alpha_i \quad (3)$$

$$0 \leq y_i \leq \beta_i, \quad \forall i = 1, \dots, n, \quad (4)$$

where $F(\cdot)$ is jointly convex in $\vec{y} = (y_1, \dots, y_n)$, and $0 \leq \alpha_i < +\infty$, $0 \leq \beta_i \leq +\infty$, for $i = 1, \dots, n$. We make the following contributions in this paper.

1. We develop a dual method to solve a special case of **(P1)** with separable objective functions and (3) being an inequality constraint. Our dual method stops in a finite number of iterations and improves the worst case complexity over the algorithm in [2].
2. Using the dual method as a subroutine, we propose a gradient projection method to solve **(P1)**. Our proposed method takes advantages of the structure of the constraints so that each projection step can be completed efficiently. The gradient projection method also allows non-separable objective functions and equality constraint in (3).

*Industrial and System Engineering, University of Minnesota, Minneapolis, 55414. Email: zzwang@umn.edu

3. We perform numerical experiments on several test problems. The test results show that our proposed algorithms outperform the algorithm in [2] as well as other standard methods in most problems.

1.1 Equivalent Form

We first point out an equivalent form of **(P1)** which is frequently used in the literature:

$$\begin{aligned}
(\mathbf{P2}) \quad & \text{minimize}_{\vec{y}} \quad G(\vec{y}) = g(y_1, \dots, y_n) \\
& \text{subject to} \quad \sum_{i=1}^k y_i \geq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n-1 \\
& \quad \quad \quad \sum_{i=1}^n y_i = (\geq) \sum_{i=1}^n \alpha_i \\
& \quad \quad \quad 0 \leq y_i \leq \beta_i \quad \forall i = 1, \dots, n,
\end{aligned} \tag{5}$$

where $G(\vec{y})$ is jointly convex in \vec{y} . To translate (5) into **(P1)**, we define $z_i = \beta_i - y_i$, and replace y_i by z_i , then the optimization problem becomes:

$$\begin{aligned}
& \text{minimize}_{\vec{z}} \quad F(\vec{z}) = G(\beta_1 - z_1, \dots, \beta_n - z_n) \\
& \text{subject to} \quad \sum_{i=1}^k z_i \leq \sum_{i=1}^k (\beta_i - \alpha_i), \quad \forall k = 1, \dots, n-1 \\
& \quad \quad \quad \sum_{i=1}^n z_i = (\leq) \sum_{i=1}^n (\beta_i - \alpha_i) \\
& \quad \quad \quad 0 \leq z_i \leq \beta_i, \quad \forall i = 1, \dots, n,
\end{aligned}$$

which is exactly of form **(P1)**.

1.2 Applications

The formulation **(P1)** arises in many applications. One example which is a problem of smoothing is discussed in [5]. Another one that arises in a special case of network flow problems is studied in [6] and [7]. Both these two examples have the form of **(P2)** with $G(\vec{y}) = \sum_i \theta_i y_i^p$. Other problems arise frequently in communications and are discussed in [2], [8] and [3]. Here we present another motivating application of this model in operations management.

Inventory problem with downward substitution. A firm sells a product with n different grades, with 1 the highest and n the lowest. The firm has α_i grade i products on hand and is facing a random demand D_i for each grade i . Any product of grade i can be used to satisfy the demand of product of grade i or lower ($j \geq i$). Before the demand realizes, the firm has to make an inventory decision y_i of how much grade i product to put into stock. Once this is done, the products are no longer substitutable (for example, the firm has to package these products during this process, products of different grades need different packages and will not be distinguishable after packaging). For each grade i , there is a unit overage cost o_i if D_i is less than y_i and a unit underage cost u_i if D_i is greater than y_i . The objective is to minimize the expected total cost. The problem can be written as:

$$\begin{aligned}
& \text{minimize}_{\vec{y}} \quad \sum_{i=1}^n (u_i E(D_i - y_i)^+ + o_i E(y_i - D_i)^+) \\
& \text{subject to} \quad \sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n \\
& \quad \quad \quad y_i \geq 0, \quad \forall i = 1, \dots, n,
\end{aligned} \tag{6}$$

which is of form **(P1)**. In practice, one might have strong incentive to solve (6) faster. For example, the firms may also need to decide the upfront production quantities α_i 's

for each grade with production cost $c_i\alpha_i$. The actual yield of grade i product is $\alpha_i U_i$ where U_i 's are some known random yield distributions. In this case, the firm's problem is the following two-stage stochastic programming problem:

$$\begin{aligned} \text{minimize}_{\vec{\alpha}} \quad & \sum_{i=1}^n c_i \alpha_i + E_U \{ \text{minimize}_{\vec{y}} \sum_{i=1}^n (u_i E(D_i - y_i)^+ + o_i E(y_i - D_i)^+) \} \\ \text{subject to} \quad & \sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i U_i, \quad \forall k = 1, \dots, n \\ & y_i \geq 0, \quad \forall i = 1, \dots, n. \end{aligned} \quad (7)$$

A natural approach to solve (7) is to use the stochastic gradient method [9]. However, this requires one to evaluate the inside problem repeatedly. Therefore, improving the efficiency of solving (6) could be of strong interest.

1.3 Literature Review

The main related literature to this paper is [2]. In [2], the authors propose a dual method for solving **(P2)** with separable objective functions. We call the algorithm in [2] the “P-S algorithm” in the rest of the discussions. The P-S algorithm finishes in $O(n)$ outer iterations. In each iteration, it solves up to n nonlinear equations, and sets at least one primal variable based on the solutions to the equations. The efficiency of the P-S algorithm depends on how fast one can solve those equations. When the equations have close form solutions, the P-S algorithm performs very well, otherwise, it may not. In this paper, we propose a dual algorithm which does not attempt to set primal variables in each iteration. Instead, we set one dual variable in each iteration and maintain the optimality conditions for the variables that have been set. Our dual algorithm also finishes in $O(n)$ outer iterations and in each iteration, we solve no more than one equation. We show that the equations we solve are simply the equations in the P-S algorithm with lower bound on each term. When the equations in P-S algorithm do not have a close form solution, solving both equations usually have the same complexity. In those cases, our dual algorithm reduces the computational complexity of the P-S algorithm by an order of n .

In addition to the dual method, we propose a gradient projection method to solve the more general problem **(P1)** allowing non-separable objective functions. Gradient projection methods are widely used to solve a variety of convex optimization problems. We refer the readers to [1] for a thorough discussion of this method. In particular, the key element in gradient projection method is the design of the projection step. To my best knowledge, this paper is the first one to study the projection step under linear ascending constraints. Therefore, the result may be of independent interest from this perspective.

Another popular method that solves nonlinear convex optimization is the interior point method. However, we focus on the first order method in this paper because of its low memory requirement and thus the ability to solve large problems. Performance comparisons between our proposed algorithms and the interior point algorithm (implemented by CVX) are shown in the numerical tests and the results indicate that our algorithms are usually much more efficient.

We note that there is abundant literature on solving a special case of **(P1)** when there is only one equality/inequality constraint (usually called the simplex constraint or

l_1 constraint). We refer the readers to [11] for a survey on this problem. Although the dual method is widely used in those studies, the detail of our algorithm differs because of the special structure of this problem. As we show later, the specific structure of the constraint is the key to make our algorithm work.

1.4 Structure of the paper

In Section 2, we develop a dual method to solve a special case of **(P1)** with separable objective functions and (3) being an inequality constraint. In Section 3, we further propose a gradient projection method to solve the general problem **(P1)**. Numerical tests are shown in Section 4 to examine the performances of our algorithms. Section 5 concludes this paper.

2 A Dual Method

In this section, we study a special case of **(P1)** in which the objective function F is separable, i.e., $F(\vec{y}) = \sum_{i=1}^n f_i(y_i)$ and (3) is an inequality constraint. There are two reasons why we consider separable objectives. First, in most of the applications mentioned in Section 1.2, the objective functions are indeed separable. Second, the study of separable objective functions will lay the foundation for the analysis of the gradient projection method in Section 3 which can solve more general problems.

In the following, we develop a dual method to solve the following problem:

$$\textbf{(P3)} \quad \text{minimize}_{\vec{y}} \quad \sum_{i=1}^n f_i(y_i) \quad (8)$$

$$\text{subject to} \quad \sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n \quad (9)$$

$$0 \leq y_i \leq \beta_i, \quad \forall i = 1, \dots, n. \quad (10)$$

Here we assume that $f_i(\cdot)$'s are continuously differentiable and define $g_i(x) = f'_i(x)$ ¹. Without loss of generality, we assume that $\beta_i \leq \sum_{k=1}^i \alpha_k$. Furthermore, we assume that $g_i(\cdot)$ is strictly increasing with $g_i(0) = l_i$ and $g_i(\beta_i) = h_i$. We define $\bar{y}_i = \arg \min_{0 \leq y \leq \beta_i} f_i(y)$, that is, \bar{y}_i 's are the optimal solution to **(P3)** without constraint (9). Under the above assumptions, it is easy to see that \bar{y}_i exists and is unique.

We first construct the KKT conditions of (8)-(10). We associate a dual variable λ_k to each constraint (9), a dual variable δ_i to each upper bound constraint, and a dual variable η_i to each nonnegative constraint (10). The Lagrangian of (8)-(10) can then be written as

$$\sum_{i=1}^n f_i(y_i) + \sum_{k=1}^n \lambda_k \left(\sum_{i=1}^k y_i - \sum_{i=1}^k \alpha_i \right) - \sum_{i=1}^n \eta_i y_i - \sum_{i=1}^n \delta_i (\beta_i - y_i).$$

¹Our algorithm works in a similar manner even if f is not differentiable but convex. The discussions will involve subgradient of f in that case. We make this assumption simply for the convenience of discussion.

And the KKT conditions are

$$g_i(y_i) = - \sum_{k=i}^n \lambda_k + \eta_i - \delta_i, \quad \forall i = 1, \dots, n, \quad (11)$$

$$y_i \cdot \eta_i = 0, y_i \geq 0, \eta_i \geq 0, \quad \forall i = 1, \dots, n, \quad (12)$$

$$(\beta_i - y_i) \cdot \delta_i = 0, y_i \leq \beta_i, \delta_i \geq 0, \quad \forall i = 1, \dots, n, \quad (13)$$

$$\sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n, \quad (14)$$

$$\lambda_k \cdot \left(\sum_{i=1}^k y_i - \sum_{i=1}^k \alpha_i \right) = 0, \lambda_k \geq 0, \quad \forall k = 1, \dots, n. \quad (15)$$

Define $\phi_i(x) = \max\{l_i, \min\{x, h_i\}\}$ and $H_i(x) = g_i^{-1}(\phi_i(x))$. By the assumptions on $g_i(\cdot)$, l_i and h_i , we have $0 \leq H_i(x) \leq \beta_i$. And conditions (11)-(13) can be equivalently written as

$$y_i = H_i \left(- \sum_{k=i}^n \lambda_k \right) \quad (16)$$

with

$$\eta_i = \left(\phi \left(- \sum_{k=i}^n \lambda_k \right) + \sum_{k=i}^n \lambda_k \right)^+ \quad \text{and} \quad \delta_i = \left(- \phi \left(- \sum_{k=i}^n \lambda_k \right) - \sum_{k=i}^n \lambda_k \right)^+,$$

where $x^+ = \max\{x, 0\}$. Since **(P3)** is linearly constrained and is convex, solving the KKT conditions are equivalent as solving the original problem [12]. In the following, we propose an efficient dual method to solve the KKT conditions. The idea of this dual method is to assign values to the dual variables λ 's such that the optimality conditions (14)-(16) hold. We state our algorithm as follows:

Algorithm 1

Step 0: Initialization. Let $d_k = \sum_{i=1}^k \bar{y}_i - \sum_{i=1}^k \alpha_i$, $k = 1, 2, \dots, n$. Define

$$\begin{aligned} w_0 &= 0, \\ w_1 &= \min\{k : d_k \geq 0\}, \\ w_{j+1} &= \min\{k > w_j : d_k \geq d_{w_j}\}. \end{aligned}$$

Here we define $\min \emptyset = \infty$. If $w_1 = \infty$, then setting $y_i = \bar{y}_i$ and $\lambda_i = 0$ for all i will satisfy the KKT conditions and thus is optimal. Otherwise, let $L = \max\{j \geq 1 : w_j < \infty\}$. Define $S = \{w_1, w_2, \dots, w_L\}$. Let $\lambda_i = 0$, $\eta_i = 0$ for all i and let $j = L$.

Step 1: Main Loop (Outer Loop).

WHILE $j > 0$

- Case 1: If

$$\sum_{s=w_{j-1}+1}^{w_j} \left(H_s \left(- \sum_{t=j+1}^L \lambda_{w_t} \right) - \alpha_s \right) \geq 0 \quad (17)$$

then choose $\xi \geq 0$ such that

$$\sum_{s=w_{j-1}+1}^{w_j} \left(H_s \left(- \sum_{t=j+1}^L \lambda_{w_t} - \xi \right) - \alpha_s \right) = 0 \quad (18)$$

and set $\lambda_{w_j} = \xi$, $j = j - 1$.

- Case 2: If

$$\sum_{s=w_{j-1}+1}^{w_j} \left(H_s \left(- \sum_{t=j+1}^L \lambda_{w_t} \right) - \alpha_s \right) < 0 \quad (19)$$

then use binary search to find

$$r^* = \min \left\{ j+1 \leq r \leq L : \sum_{s=w_{j-1}+1}^{w_r} \left(H_s \left(- \sum_{t=r+1}^L \lambda_{w_t} \right) - \alpha_s \right) \geq 0 \right\}. \quad (20)$$

If such r^* does not exist, then set all $\lambda_{w_r} = 0$, for $r = j+1, \dots, L$ and $j = j - 1$. Otherwise, choose $\xi \geq 0$ such that

$$\sum_{s=w_{j-1}+1}^{w_{r^*}} \left(H_s \left(- \sum_{t=r^*+1}^L \lambda_{w_t} - \xi \right) - \alpha_s \right) = 0 \quad (21)$$

and set $\lambda_{w_{r^*}} = \xi$ and $\lambda_{w_r} = 0$, for $r < r^*$. Set $j = j - 1$.

END WHILE

Step 2: Output Set

$$y_i = H_i \left(- \sum_{k=i}^n \lambda_k \right),$$

$$\eta_i = \left(\phi \left(- \sum_{k=i}^n \lambda_k \right) + \sum_{k=i}^n \lambda_k \right)^+ \quad \text{and} \quad \delta_i = \left(-\phi \left(- \sum_{k=i}^n \lambda_k \right) - \sum_{k=i}^n \lambda_k \right)^+.$$

First, we argue that those ξ 's defined in (18) and (21) exist. This can be verified by observing that when $\xi = 0$, the left hand sides of (18) and (21) are both nonnegative and as $\xi \rightarrow \infty$, both of them will be less than zero. Also, by our assumption, the left hand sides of (18) and (21) are both continuous. Therefore, by the intermediate value theorem, such ξ 's must exist. We now state the main result of this section.

Theorem 1 *Algorithm 1 terminates within $L \leq n$ outer iterations and the output minimizes (8)-(10).*

In the following, we prove Theorem 1. Let $(\{y_i^*\}_{i=1}^n, \{\lambda_i^*\}_{i=1}^n, \{\eta_i^*\}_{i=1}^n, \{\delta_i^*\}_{i=1}^n)$ be any solution to the KKT conditions (14)-(16) and thus an optimal solution to the original problem. First, it is easy to see that $y_i^* \leq \bar{y}_i$ for all i , otherwise replace y_i^* with \bar{y}_i will strictly improve the objective value while still satisfying the constraints. Next we claim that for any $k \notin S$, we must have $\lambda_k^* = 0$. This is because for $w_{l-1} < k < w_l$, we have

$$\sum_{i=1}^k (y_i^* - \alpha_i) \leq \sum_{i=w_{l-1}+1}^k (y_i^* - \alpha_i) \leq \sum_{i=w_{l-1}+1}^k (\bar{y}_i - \alpha_i) < 0,$$

where the last inequality is because of the definition of w_l . By the complementarity condition (15), $\lambda_k^* = 0$ for $w_{l-1} < k < w_l$.

Note that in the KKT conditions, given λ_k 's, the y 's, η 's and δ 's are uniquely determined and that changing λ_k only affects y_i 's, η_i 's and δ_i 's for $i \leq k$. In each iteration of Algorithm 1, we assign one new λ_{w_l} and may modify all λ_{w_k} 's for $k \geq l+1$. We now state the following property of Algorithm 1 which immediately implies Theorem 1.

Proposition 1 *When Algorithm 1 finishes loop j ($j = L, L-1, \dots, 1$), the current λ_i 's together with*

$$y_i = H_i \left(- \sum_{k=i}^n \lambda_k \right), \quad (22)$$

$$\eta_i = \left(\phi \left(- \sum_{k=i}^n \lambda_k \right) + \sum_{k=i}^n \lambda_k \right)^+ \quad \text{and} \quad \delta_i = \left(- \phi \left(- \sum_{k=i}^n \lambda_k \right) - \sum_{k=i}^n \lambda_k \right)^+ \quad (23)$$

satisfy the following conditions:

$$y_i \cdot \eta_i = 0, \eta_i \geq 0, y_i \geq 0, \quad \forall i \geq w_{j-1} + 1, \quad (24)$$

$$(\beta_i - y_i) \cdot \delta_i = 0, y_i \leq \beta_i, \delta_i \geq 0, \quad \forall i \geq w_{j-1} + 1, \quad (25)$$

$$\sum_{s=w_{j-1}+1}^k y_s \leq \sum_{s=w_{j-1}+1}^k \alpha_s, \quad \forall k \geq w_{j-1} + 1, \quad (26)$$

$$\lambda_k \cdot \left(\sum_{s=w_{j-1}+1}^k y_s - \sum_{s=w_{j-1}+1}^k \alpha_s \right) = 0, \quad \forall k \geq w_{j-1} + 1. \quad (27)$$

Before we prove Proposition 1, we introduce a lemma that will be used repeatedly in the proof.

Lemma 1 *If y_i is defined in (22) for some nonnegative λ_k 's, then $y_i \leq \bar{y}_i$.*

The lemma follows immediately from the assumption that $g_i(\cdot)$'s are strictly increasing and that $\bar{y}_i = H_i(0)$.

Proof of Proposition 1: First, note that condition (24) and (25) are satisfied for all j 's because of the definitions in (22) and (23). Therefore, it suffices to show that conditions (26) and (27) hold for $j = L, L-1, \dots, 1$. We use backward induction for this. First we show that for $j = L$, (26) and (27) hold for all $k \geq w_{L-1} + 1$.

First we show that (26) holds. When Algorithm 1 finishes loop L , for any $w_{L-1} + 1 \leq s' < w_L$, we have

$$\sum_{s=w_{L-1}+1}^{s'} (y_s - \alpha_s) \leq \sum_{s=w_{L-1}+1}^{s'} (\bar{y}_s - \alpha_s) \leq 0, \quad (28)$$

where the first inequality is due to Lemma 1 and the second inequality is due to the definition of w_L . On the other hand, for $s' \geq w_L$, we have

$$\sum_{s=w_{L-1}+1}^{s'} (y_s - \alpha_s) \leq \sum_{s=w_{L-1}+1}^{w_L} (y_s - \alpha_s) + \sum_{s=w_L+1}^{s'} (\bar{y}_s - \alpha_s) \leq 0,$$

where the first inequality is because of Lemma 1 and the second one is because of the definition of w_L . Therefore (26) holds when $j = L$.

To show that (27) holds for $j = L$, note that among all the λ_k 's with $k \geq w_{L-1} + 1$, the only possible non-zero one is λ_{w_L} . If Case 1 of the algorithm happens in this loop, then

$$\sum_{s=w_{L-1}+1}^{w_L} (y_s - \alpha_s) = 0.$$

Otherwise, $\lambda_{w_L} = 0$. Therefore, (27) holds for $j = L$.

Now we assume that (26) - (27) hold after the algorithm completes the outer loop for $j = \bar{j} + 1$. Now we consider the situation when it finishes the outer loop for $j = \bar{j}$. We consider two cases:

- Case 1: (17) holds in the current loop ($j = \bar{j}$). In this case, we have

$$\sum_{s=w_{\bar{j}-1}+1}^{w_{\bar{j}}} y_s = \sum_{s=w_{\bar{j}-1}+1}^{w_{\bar{j}}} \alpha_s. \quad (29)$$

And the y_s 's for $s > w_{\bar{j}}$ does not change from the previous loop. Therefore, for any $k = w_{\bar{j}}$ ($j \geq \bar{j}$), we have

$$\sum_{s=w_{\bar{j}-1}+1}^k y_s \leq \sum_{s=w_{\bar{j}-1}+1}^k \alpha_s.$$

And for $w_{\bar{j}} < k < w_{\bar{j}+1}$ ($j \geq \bar{j} - 1$),

$$\sum_{s=w_{\bar{j}-1}+1}^k (y_s - \alpha_s) \leq \sum_{s=w_{\bar{j}-1}+1}^{w_{\bar{j}}} (y_s - \alpha_s) + \sum_{s=w_{\bar{j}}+1}^k (\bar{y}_s - \alpha_s) \leq \sum_{s=w_{\bar{j}-1}+1}^{w_{\bar{j}}} (y_s - \alpha_s) \leq 0,$$

where the first inequality is because of Lemma 1 and the second inequality is because of the definition of w_j 's. Therefore, (26) holds for all $k \geq w_{\bar{j}-1} + 1$. For (27), we only need to study $k = w_j$ since all other λ_k 's are 0. And it holds because of (29) and the induction assumption. Therefore, (26) - (27) hold for \bar{j} in this case.

- Case 2: (19) holds in the current loop. Then there are two further cases:
 - a): r^* does not exist. In this case, by the definition of Algorithm 1, all λ_k 's are zero after this iteration and $\sum_{s=w_{\bar{j}-1}+1}^k y_s < \sum_{s=w_{\bar{j}-1}+1}^k \alpha_s$ for all $k = w_j$ ($j \geq \bar{j}$). By the same argument as in case 1, we know that $\sum_{s=w_{\bar{j}-1}+1}^k y_s < \sum_{s=w_{\bar{j}-1}+1}^k \alpha_s$ for all $k \geq w_{\bar{j}-1} + 1$. Therefore, (26) - (27) hold for \bar{j} in this case.
 - b): r^* exists. Denote the λ 's and y 's after the previous loop by $\tilde{\lambda}$ and \tilde{y} . It is easy to see that in this case, $\lambda_i \leq \tilde{\lambda}_i$ and $y_i \geq \tilde{y}_i$ for all i . We first show the following lemma whose proof is referred to Appendix A:

Lemma 2 $\tilde{\lambda}_{w_{r^*}} > 0$.

With Lemma 2, we show that (26) - (27) hold. We first consider (26). By (21), we have

$$\sum_{s=w_{\bar{j}-1}+1}^{w_{r^*}} (y_s - \alpha_s) = 0. \quad (30)$$

Therefore for $k > w_{r^*}$, we know that

$$\sum_{s=w_{\bar{j}-1}+1}^k (y_s - \alpha_s) = \sum_{s=w_{r^*}+1}^k (y_s - \alpha_s) = \sum_{s=w_{r^*}+1}^k (\tilde{y}_s - \alpha_s) = \sum_{s=w_{\bar{j}}+1}^k (\tilde{y}_s - \alpha_s) \leq 0,$$

where the second equality is because that y_s does not change for $s \geq w_{r^*} + 1$. The last equality is because of the induction assumption that $\tilde{\lambda}_{w_{r^*}} \cdot \sum_{s=w_{\bar{j}}+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s) = 0$ and Lemma 2. Therefore, for all $w_j \leq k < w_{j+1}$, $\bar{j} - 1 \leq j < r^*$, we have

$$\sum_{s=w_{\bar{j}-1}+1}^k (y_s - \alpha_s) \leq \sum_{s=w_{\bar{j}-1}+1}^{w_j} (y_s - \alpha_s) \leq - \sum_{s=w_j+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s) = \sum_{s=w_{\bar{j}}+1}^{w_j} (\tilde{y}_s - \alpha_s) \leq 0,$$

where the first inequality is because of the definition of w_j , the second equality is because of (30) and $y_i \geq \tilde{y}_i$ and the last equality is because of the induction assumption and Lemma 2. Therefore, (26) holds in this case.

Lastly, we show that (27) also holds. It suffices to show that for each $r > r^*$ such that $\lambda_r > 0$,

$$\sum_{s=w_{\bar{j}-1}+1}^{w_r} (y_s - \alpha_s) = 0.$$

This is equivalent as showing that for each $r > r^*$ such that $\lambda_r > 0$,

$$\sum_{s=w_{r^*}+1}^{w_r} (y_s - \alpha_s) = 0.$$

Note that

$$\sum_{s=w_{r^*}+1}^{w_r} (y_s - \alpha_s) = \sum_{s=w_{r^*}+1}^{w_r} (\tilde{y}_s - \alpha_s) = \sum_{s=w_{\bar{j}}+1}^{w_r} (\tilde{y}_s - \alpha_s) - \sum_{s=w_{\bar{j}}+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s)$$

By induction assumption and Lemma 2,

$$\sum_{s=w_{\bar{j}}+1}^{w_r} (\tilde{y}_s - \alpha_s) = \sum_{s=w_{\bar{j}}+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s) = 0.$$

Therefore (27) holds in this case and Proposition 1 is proved. \square

Now we make some comments on Algorithm 1.

By its definition, Algorithm 1 terminates within $L \leq n$ outer iterations. In practical problems, L might be much less than n . In those cases, the algorithm can output the solutions very fast. This is a similar property as in the P-S algorithm (recall we use P-S algorithm to refer the algorithm proposed in [2]). Now we use \mathcal{I} to denote the complexity (number of arithmetic operations) of solving (18) or (21) once (it is easy to see that $\mathcal{I} \geq n$). In each iteration of Algorithm 1, if Case 1 happens, the algorithm has to perform a sum of no more than n terms. And it has to solve (18) once. Therefore, there are $O(\mathcal{I})$ arithmetic operations in this case. If Case 2 happens, then the algorithm has similar tasks as in Case 1, and in addition it needs to find r^* defined in (20) which takes no more than $O(n \log n)$ iterations. Therefore, the complexity in Case 2 is $O(\max\{n \log n, \mathcal{I}\})$. Combined with $O(n)$ outer iterations, the total arithmetic complexity of our algorithm is $O(\max\{n^2 \log n, n\mathcal{I}\})$.

Now we compare the complexity result to that of the P-S algorithm. The difference between the two algorithms is the way the variables are assigned. In each iteration of the P-S algorithm, it solves

$$\sum_{m \in s \cap [i, l]} (g_m^{-1}(\theta) \wedge \beta_m) = \sum_{m=i}^l \alpha_m \quad (31)$$

for all $l \leq j$, where s is the set of unassigned variables. Then the largest solution is chosen and the corresponding primal variable is set accordingly. Such a method avoids the needs to check the validity of the KKT conditions that is met in previous steps as we have to do in Step 2 in Algorithm 1, however at a cost of having to solve $O(n)$ equations at each step rather than only one as in Algorithm 1. Indeed, the equations (31) are sometimes easier to solve since they don't involve the lower bound as Algorithm 1 do. If one denotes the arithmetic complexity of solving equations in (31) by $O(\mathcal{I}')$, then the total arithmetic complexity of the P-S algorithm is $O(n^2 \mathcal{I}')$. Therefore, our algorithm works better than the P-S algorithm when solving equations in (31) has similar complexity as solving equations in (18) and (21), but may work relatively worse if (31) can be solved explicitly (see [2] for several examples). This tradeoff is demonstrated in the numerical experiments in Section 4.

There are two main drawbacks for Algorithm 1. First, it can only handle separable objective functions and inequality constraint in (3). Second, it involves many evaluations of g^{-1} and also has to solve the equations (18) and (21). These evaluations might

be very expensive in computation if g^{-1} does not have a simple form. This is the same problem as in the P-S algorithm. In particular, [2] shows that the performance of the P-S algorithm may not be very good if close form solutions to equations (31) do not exist. To overcome this problem, we propose a gradient projection method in the next section. The gradient projection method uses Algorithm 1 as a subroutine, however, in each iteration, $g(\cdot)$ is simply a linear function. Moreover, the gradient projection method can handle non-separable objective functions as well as equality constraints in (3). The tradeoff however, is that the gradient projection method does not give an exact solution in a finite number of iterations. However as we demonstrate in our numerical experiments, it performs quite well in test problems.

3 Gradient Projection Method

In this section, we propose a gradient projection method to solve (P1). First we claim that we can assume that constraint (3) is of the inequality form. To transform a problem with equality constraint in (3) to an inequality one, we first note that we can without loss of generality assume $\beta_n = \infty$. This is because one can always add a penalty term $M(y_n - \beta_n)^+$ with sufficiently large M so that the optimal solution must satisfy $y_n \leq \beta_n$ (if the problem is feasible). Then, we can simply substitute $y_n = \sum_{i=1}^n \alpha_i - \sum_{i=1}^{n-1} y_i$ into (8). Therefore, it is sufficient to consider the following equivalent problem:

$$\begin{aligned}
(\mathbf{P4}) \quad & \text{minimize}_{\vec{y}} \quad F(\vec{y}) = f(y_1, \dots, y_n) \\
& \text{subject to} \quad \sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n \\
& \quad \quad \quad 0 \leq y_i \leq \beta_i, \quad \forall i = 1, \dots, n.
\end{aligned} \tag{32}$$

In the following, we propose a gradient projection method to solve (P4). Gradient projection methods are used to solve a variety of convex optimization problems [1]. It minimizes a function $F(x)$ subject to convex constraints by generating the sequence $\{x^{(k)}\}$ via

$$x^{(k+1)} = x^{(k)} - \eta_k \Pi^k \left(\nabla^{(k)} \right),$$

where $\nabla^{(k)}$ is the (sub)gradient of $F(x)$ at $x^{(k)}$, $\Pi^k(x) = \arg \min_y \{ \|x - y\| : y \in \mathcal{F}^{(k)} \}$ is the Euclidean projection of x onto the feasible direction space $\mathcal{F}^{(k)}$ at $x^{(k)}$ and η_k is a chosen step size. In order for such methods to work efficiently, one needs to find an efficient way to calculate $\Pi^{(k)}(x)$ for given x . Our discussion will mainly focus on the projection step, the discussion of the outer iterations of the projection gradient method is referred to [1].

Assume that for a feasible solution y , the binding constraint set of (32) is $S(y)$, that is, $\sum_{i=1}^k y_i = \sum_{i=1}^k \alpha_i$ for $k \in S(y)$, the active nonnegative constraint set of (32) is $T(y)$, that is, $y_k = 0$, for $k \in T(y)$ and the active upper bound constraint set of (32) is $R(y)$, that is, $y_k = \beta_k$, for $k \in R(y)$. Then the Euclidean projection of a descent

direction v of F at y onto the feasible direction space can be computed by:

$$\begin{aligned}
& \text{minimize}_{\vec{x}} \quad \|x - v\|_2^2 \\
& \text{subject to} \quad \sum_{i=1}^k x_i \leq 0, \quad \forall k \in S(y) \\
& \quad \quad \quad x_i \geq 0, \quad \forall i \in T(y) \\
& \quad \quad \quad x_i \leq 0, \quad \forall i \in R(y).
\end{aligned} \tag{33}$$

Now we show how (33) can be transformed into a problem of form **(P3)**. Suppose x^* is the optimal solution to (33). It is easy to see that $\|x^* - v\|_2^2 \leq \|v\|_2^2$ since $x = 0$ is a feasible solution. Therefore, $x_i^* \geq v_i - \|v\|_2$ for all i . Define

$$z_i = \begin{cases} x_i - v_i + \|v\|_2 & i \notin T(y) \\ x_i & i \in T(y). \end{cases}$$

and

$$\tilde{v}_i = \begin{cases} \|v\|_2 & i \notin T(y) \\ v_i & i \in T(y). \end{cases}$$

Then we can rewrite (33) as

$$\begin{aligned}
& \text{minimize}_{\vec{z}} \quad \sum_{i=1}^n (z_i - \tilde{v}_i)^2 \\
& \text{subject to} \quad \sum_{i=1}^k z_i \leq \gamma_k, \quad \forall k = 1, \dots, n \\
& \quad \quad \quad 0 \leq z_i \leq \nu_i, \quad \forall i = 1, \dots, n,
\end{aligned} \tag{34}$$

where

$$\gamma_k = \min \left\{ n\|v\|_2 + \|v\|_1, \min \left\{ \tilde{k} \geq k, \tilde{k} \in S(y) : \sum_{i=1, i \in S(y)}^{\tilde{k}} (\|v\|_2 - v_i) \right\} \right\},$$

$$\nu_i = \begin{cases} \|v\|_2 - v_i & i \in R(y) \\ +\infty & i \notin R(y). \end{cases}$$

Note that (34) is of form **(P3)** thus can be solved by Algorithm 1. One main advantage of (34) is that the objective function is quadratic. Therefore, in Algorithm 1, $g_i(z_i) = 2(z_i - \tilde{v}_i)$, $l_i = -2\tilde{v}_i$, $h_i = 2(\nu_i - \tilde{v}_i)$ and $g_i^{-1}(u_i) = \frac{u_i + 2\tilde{v}_i}{2}$. Therefore the equation (18) and similarly (21) can be written as

$$\rho(\xi) = \sum_{s=w_{j-1}+1}^{w_j} \left(\frac{\max(0, \min(2\nu_i, 2\tilde{v}_s - \sum_{t=j+1}^L \lambda_{w_t} - \xi))}{2} - \alpha_s \right) = 0. \tag{35}$$

Note that (35) is a decreasing piecewise linear function with no more than $2n$ breakpoints. And those breakpoints can be computed explicitly. Therefore, to solve (35), one can first use binary search to find out which piece of the function the solution belongs to and then simply solve a linear equation. Therefore, the total complexity of solving (35) is $O(n \log n)$ and the total complexity of each projection step is $O(n^2 \log^2 n)$, regardless of the form of the objective function.

#	Problem	n	DM	GP	CVX	P-S
1	(TP-1)	50	0.050	0.214	0.378	0.154
2	(TP-1)	150	0.370	0.681	1.792	3.751
3	(TP-1)	500	5.614	2.559	15.76	219.8
4	(TP-1)	2000	242.3	34.58	4953.1	N/A
5	(TP-2)	50	0.018	0.093	0.264	< 0.001
6	(TP-2)	150	0.136	0.176	0.547	< 0.001
7	(TP-2)	500	1.911	0.806	14.04	0.0013
8	(TP-2)	2000	58.15	3.160	4798.9	0.0026
9	(TP-3)	50	0.011	0.088	1.001	0.227
10	(TP-3)	150	0.015	0.124	4.950	0.966
11	(TP-3)	500	0.210	0.374	49.44	6.210
12	(TP-3)	2000	0.499	1.526	2350.2	91.66

Table 1: Performance Comparisons. DM is the dual method developed in Section 2, GP is the gradient projection method developed in Section 3 and P-S is the algorithm in [2]. N/A means this method can not return the optimal solution in the corresponding case

4 Numerical Experiments

In this section, we perform numerical tests to examine the performance of both our dual method and the gradient projection method and compare them to 1) the P-S algorithm in [2] and 2) CVX [10]. The P-S algorithm also uses a dual method and the comparison between it and Algorithm 1 is discussed in Section 2. CVX is a popular convex optimization solver which uses a core solver SDPT3 or Sedumi to solve a large class of convex optimization problem. It is based on interior point methods. In the following, we consider three sets of problems. For each one, we test 30 random instances with input specified in the following (for problem with size $n = 2000$, we only test 3 instances). Note that the default precision of CVX is $\epsilon = 1.5 \times 10^{-8}$. In our dual method, we solve each equation with precision ϵ , and for the gradient method, our stopping criterion is that the objective is within ϵ to the CVX optimal value. All the computations are run on a PC with 1.80GHz CPU and Windows 7 Operating system. We use CVX Version 1.22 and MATLAB version R2010b. The test results are shown in Table 1.

The first problem is

$$\begin{aligned}
(\mathbf{TP} - 1) \quad & \text{minimize} \quad \sum_{i=1}^n \left(\frac{1}{4} y_i^4 + v_i y_i \right) \\
& \text{subject to} \quad \sum_{i=1}^k y_i \geq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n-1 \\
& \quad \quad \quad \sum_{i=1}^n y_i = \sum_{i=1}^n \alpha_i \\
& \quad \quad \quad y_i \geq 0, \quad \quad \quad \forall i = 1, \dots, n.
\end{aligned} \tag{36}$$

This problem is considered in the numerical tests in [2]. We use the same setup, that is, we assume all the parameters α_i and v_i are drawn from i.i.d. uniform distributions on $[0, 1]$, with v_i 's sorted in ascending order. In order to apply our dual method and gradient projection method, we first perform a transformation as described in Section 1.1. Note that the equality constraint in (36) can be replaced by an inequality

constraint since the objective of (36) is increasing in y . Then we add an artificial upper bound $\beta = \sum_{i=1}^n \alpha_i$ to all y_i 's and define $z_i = \beta - y_i$. After these transformations, the problem becomes

$$\begin{aligned} & \text{minimize}_{\bar{z}} \quad \sum_{i=1}^n f_i(z_i) = \sum_{i=1}^n \left(\frac{1}{4}(\beta - z_i)^4 + v_i(\beta - z_i) \right) \\ & \text{subject to} \quad \sum_{i=1}^k z_i \leq \sum_{i=1}^k (\beta - \alpha_i), \quad \forall k = 1, \dots, n \\ & \quad \quad \quad z_i \geq 0, \quad \forall i = 1, \dots, n, \end{aligned}$$

which can be solved by both our dual method and the gradient projection method.

From Table 1, we can see that both our algorithms outperform the P-S algorithm and CVX for this problem. The reason that the dual algorithms performs better than the P-S algorithm is explained in Section 2. Particularly, in this case, $g_i^{-1}(x) = (x - v_i)^{1/3}$ and the equation (31) does not have a close form solution. In such cases, the complexity of solving (31) is essentially similar to the complexity of solving (18) or (21). And our dual algorithm is faster than the P-S algorithm in an order of n .

Note that in this problem

$$\bar{z}_i = \arg \min_{0 \leq z \leq \beta} f_i(z) = \beta.$$

Therefore $L = n$, that means this is already the worst case scenario for the dual method. Yet it still performs quite well. Because of the same reason, the gradient projection method works better than the dual method in this case. And both of them perform better than CVX significantly.

The second problem is

$$\begin{aligned} (\mathbf{TP} - 2) \quad & \text{minimize}_{\bar{y}} \quad \sum_{i=1}^n \frac{v_i}{1 - y_i} \\ & \text{subject to} \quad \sum_{i=1}^k y_i \geq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n-1 \\ & \quad \quad \quad \sum_{i=1}^n y_i = \sum_{i=1}^n \alpha_i \\ & \quad \quad \quad 0 \leq y_i \leq 1, \quad \forall i = 1, \dots, n. \end{aligned} \tag{37}$$

This problem is also considered in [2]. We again use the same setup, where α_i and v_i are drawn from i.i.d. uniform distributions on $[0, 1]$, with v_i 's sorted in ascending order. Similar to what we have done for problem 1, we replace the equality constraint with inequality and define $z_i = 1 - y_i$. An equivalent form of (37) is then obtained as follows:

$$\begin{aligned} & \text{minimize}_{\bar{z}} \quad \sum_{i=1}^n f_i(z_i) = \sum_{i=1}^n \frac{v_i}{z_i} \\ & \text{subject to} \quad \sum_{i=1}^k z_i \leq k - \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n \\ & \quad \quad \quad 0 \leq z_i \leq 1, \quad \forall i = 1, \dots, n. \end{aligned}$$

In this case, $g_i^{-1}(x) = \sqrt{-v_i/x}$. As shown in [2], there is a close form solution to (31) in this case. Thus the P-S algorithm can solve this problem very fast. This is indeed observed in Table 1. Note that the performance of both the dual and gradient projection methods also improve. This is partly because it is easier to evaluate the function g^{-1} in this case than in the first problem. Still, the gradient projection method is faster than the dual method in this case, because we have $\bar{z}_i = 1$ thus $L = n$ in the dual method. Again, both algorithms work much faster than CVX.

The last problem is the inventory control problem described in Section 1.2. The optimization problem is:

$$\begin{aligned}
(\mathbf{TP} - \mathbf{3}) \quad & \text{minimize}_{\bar{y}} \quad \sum_{i=1}^n (u_i E(D_i - y_i)^+ + o_i E(y_i - D_i)^+) \\
& \text{subject to} \quad \sum_{i=1}^k y_i \leq \sum_{i=1}^k \alpha_i, \quad \forall k = 1, \dots, n \\
& \quad \quad \quad y_i \geq 0, \quad \forall i = 1, \dots, n.
\end{aligned}$$

In the numerical experiments, we assume that $o_i \sim U[5, 10]$, $u_i \sim U[20, 25]$ and $\alpha_i \sim U[0, 20]$. We also assume that each D_i follows an exponential distribution with parameter $\eta_i \sim U[0.1, 0.2]$. By applying the property of exponential distribution, the objective can be equivalently written as:

$$\sum_{i=1}^n f_i(y_i) = \sum_{i=1}^n \left(\frac{u_i + o_i}{\eta_i} e^{-\eta_i y_i} + o_i y_i \right)$$

with $\bar{y}_i = \frac{1}{\eta_i} \log \left(\frac{u_i + o_i}{o_i} \right) \in [5.49, 17.92]$, $g_i(y_i) = f'(y_i) = -(u_i + o_i)e^{-\eta_i y_i} + o_i$ and $g_i^{-1}(s) = -\frac{1}{\eta_i} \log \left(\frac{o_i - s}{u_i + o_i} \right)$. Clearly (31) doesn't have a close form solution with such g^{-1} , therefore our algorithms outperform the P-S algorithm. In fact, for this problem, the dual method works very well. This is because the L 's in this case are usually much smaller than n . In fact, L is less than $n/4$ in most test problems. This will greatly reduce the computations in the dual method and make it very efficient. The gradient method could not take advantage of this structure and thus only has similar performance as in other problems.

To summarize the numerical results, we observe that our algorithms exhibit significant performance improvement over the P-S algorithm when equation (31) does not have a close form solution. And they also greatly improve over the performance of CVX. Between the dual method and the gradient projection method, the former one is more efficient when L is relatively small, otherwise, the latter one is usually more efficient.

5 Conclusions

In this paper, we propose two algorithms for solving a class of convex optimization problems with linear ascending inequality constraints. When the objective is separable, we propose a dual method which improves the worst case complexity of the algorithm proposed in [2]. Furthermore, we propose a gradient projection algorithm in which each projection step uses the dual method as a subroutine. The gradient projection algorithm can be used to solve more general non-separable problems and does not need to evaluate the inverse of the gradient function which the dual methods usually require. Numerical results show that both of our proposed algorithm work well in test problems.

6 Acknowledgement

I thank Diwakar Gupta and Shiqian Ma for useful discussions and Arun Padakandla and Rajesh Sundaresan for sharing the code of the P-S algorithm with me.

A Proof of Lemma 2

Proof of Lemma 2. We prove by contradiction. If $\tilde{\lambda}_{w_r^*} = 0$, then

1. If there exists $r' = \max\{0 < r < r^* : \tilde{\lambda}_{w_r} > 0\}$, then we know that

$$\sum_{s=w_{r'}+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s) \leq 0.$$

Therefore, since $r' < r^*$, we know that

$$\sum_{s=w_{\bar{j}-1}+1}^{w_{r'}} \left(g_s^{-1} \left(\max(h_s, -\sum_{t=r'+1}^L \tilde{\lambda}_{w_t}) \right) - \alpha_s \right) < 0.$$

On the other hand, we have

$$\begin{aligned} & \sum_{s=w_{\bar{j}-1}+1}^{w_{r^*}} \left(g_s^{-1} \left(\max(h_s, -\sum_{t=r^*+1}^L \tilde{\lambda}_{w_t}) \right) - \alpha_s \right) \\ = & \sum_{s=w_{\bar{j}-1}+1}^{w_{r'}} \left(g_s^{-1} \left(\max(h_s, -\sum_{t=r'+1}^L \tilde{\lambda}_{w_t}) \right) - \alpha_s \right) \\ & + \sum_{s=w_{r'}+1}^{w_{r^*}} \left(g_s^{-1} \left(\max(h_s, -\sum_{t=r^*+1}^L \tilde{\lambda}_{w_t}) \right) - \alpha_s \right) \\ = & \sum_{s=w_{\bar{j}-1}+1}^{w_{r'}} \left(g_s^{-1} \left(\max(h_s, -\sum_{t=r'+1}^L \tilde{\lambda}_{w_t}) \right) - \alpha_s \right) \\ & + \sum_{s=w_{r'}+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s) < 0. \end{aligned} \tag{38}$$

Here the first equality is because of the assumption that $\tilde{\lambda}_{w_r} = 0$ for all $r' < r \leq r^*$, and the second equality is because of the induction assumption. However, (38) contradicts with the definition of r^* .

2. If all $\tilde{\lambda}_{w_r} = 0$ for $r < r^*$. Then we have

$$\sum_{s=w_{\bar{j}-1}+1}^{w_{r^*}} (\tilde{y}_s - \alpha_s) = \sum_{s=w_{\bar{j}-1}+1}^{w_{\bar{j}}} (\tilde{y}_s - \alpha_s) + \sum_{s=w_{\bar{j}}}^{w_{r^*}} (\tilde{y}_s - \alpha_s) < 0,$$

where in the last inequality, the first term is less than 0 since the algorithm enters Case 2 in this loop, and the second term is less than or equal to 0 due to the induction assumption. Therefore we have proved Lemma 2. \square

References

- [1] D. Bertsekas. Nonlinear Programming. Athena Scientific, Belmont, MA, 2003.
- [2] A. Padakandla and R. Sundaresan. Separable Convex Optimization Problems with Linear Ascending Constraints. SIAM Journal on Optimization, 20(3): 1185-1204, 2009.
- [3] A. Padakandla and R. Sundaresan. Power Minimization for CDMA under Colored Noise. IEEE Transactions on Communications, 57(10): 3103-3112. 2009.

- [4] G. Morton, R. von Randow and K. Ringwald. A greedy algorithm for solving a class of convex programming problems and its connection with polymatroid theory. *Mathematic Programming*, 32: 238-241, 1985.
- [5] R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, NJ, 1962.
- [6] G. Dantzig. A control problem of Bellman. *Management Science*, 17: 542-546, 1971.
- [7] A. Veinott. Least d-majorized network flows with inventory and statistical applications. *Management Science*, 17: 547-567, 1971.
- [8] P. Viswanath and V. Anantharan. Optimal sequences for CDMA with colored noise: A Schur-saddle function property. *IEEE Transactions on Information Theory*, 48: 1295-1318, 2002.
- [9] A. Shapiro, D. Dentcheva and A. Ruszczycki. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, Philadelphia, PA, 2009.
- [10] M. Grant and S. Boyd. CVX: Matlab Software for Disciplined Convex Programming (Web Page and Software). <http://stanford.edu/~boyd/cvx>, 2008.
- [11] M. Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operations Research*, 185:1-46, 2008.
- [12] S. Boyd and L. Vandenberghe. *Convex Optimization* Cambridge University Press, 2004.